SUPERVISED MACHINE LEARNING: REGRESSION AND CLASSIFICATION

MACHINE LEARNING SPECIALIZATION

LINEAR REGRESSION

- Linear Regression Model or **Hypothesis** fits a straight line function

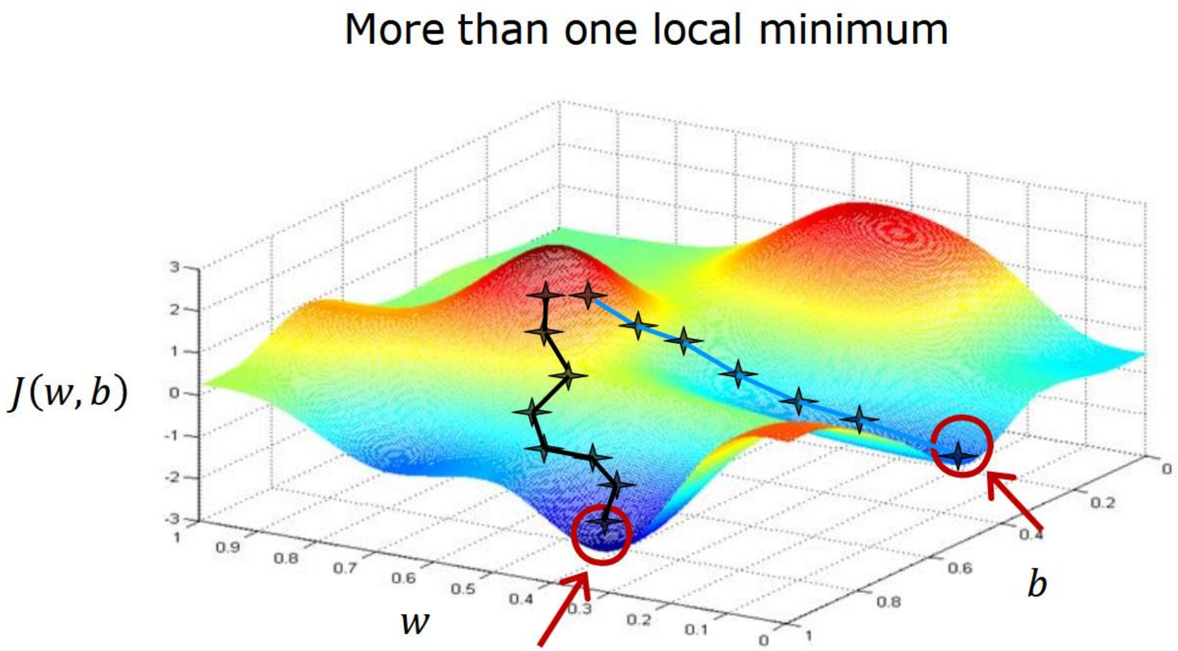
$$f_{(w,b)}(X) = wX + b$$

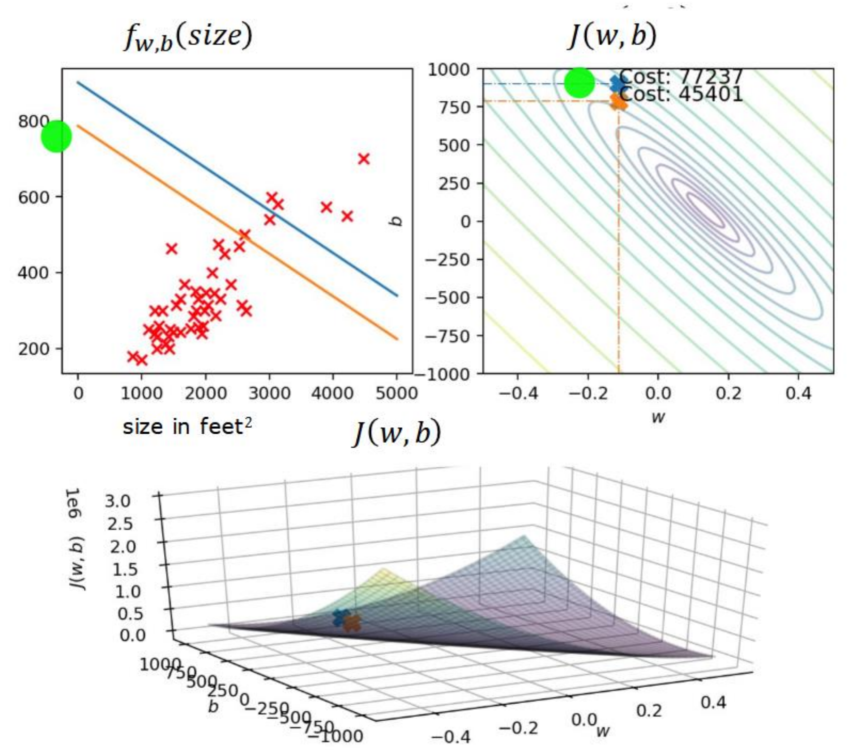
where 'X' is the input training data, 'w' is the weight parameter and 'b' is the bias constant

- Linear Regression Model running on one variable input data is **Univariate Linear Regression**
- **Training dataset 'X' has total 'm' data points** where $\boldsymbol{i^{th}}$ data point denoted as $\boldsymbol{(x^i, y^i)}$
- Linear Regression Model optimization finds weight 'w' and bias 'b' value that yields hypothesis value $\hat{y} = f_{(w,b)}(X) = wX + b$ close to original output 'Y'.
- **Cost Function (Sum of Squared Error):** $J_{(w,b)} = \frac{\Sigma(\hat{y} - y)^2}{2m}$ calculates the prediction error
- Goal of Linear Regression Model is to minimize error *J(w,b)* which means finding 'w' and 'b' that reaches **global minima** of the squared error cost function (typically **convex function**)
- Cost Functions are chosen such that it is **convex** or has only **one global minima**

LINEAR REGRESSION

- **Contour Plot** shows the cost function J in terms of 'w' and 'b' to **visualize minimum error**
- Contour Plot is **not feasible** for linear regression with **higher number of 'w' parameter**





GRADIENT DESCENT

- **Gradient Descent: Starting with 'w' and 'b' value zero**, the **optimization algorithm** iteratively minimize cost function J to reach the global minima of cost function. Cost function can have **multiple local minima** which leads different path to valley from hill. Each step updates 'w' and 'b' with **cost function derivative** which works as **slope or direction** (left or right) to reach minima from hill. Run Gradient Descent until weight parameters converge or does not update much.
- Cost Function Derivates:

$$w = w - \alpha \frac{\partial}{\partial w} J_{(w,b)} = w - \frac{\alpha}{m} \Sigma(f^i_{(w,b)} - y^i)x^i$$

$$b = b - \alpha \frac{\partial}{\partial b} J_{(w,b)} = b - \frac{\alpha}{m} \Sigma(f^i_{(w,b)} - y^i)$$

- α is the **learning rate** provides the **fixed** step size while climbing down from hill
- **Very small learning** rate α needs **more iteration** to reach minimum J
- **Very large learning** rate α **overshoot** minima or **diverge** from minima
- **Batch Gradient Descent**: Every step of iteration utilizes all input data points
- **Mini Batch Gradient Descent**: Every step of iteration utilizes small amount of input data

MULTIPLE LINEAR REGRESSION

- Multiple Linear Regression performs regression on input data with multiple parameters or features which means the input data is multi dimensional
- $x_j = j^{th}$ feature of input data X, $x^i = i^{th}$ example of input data X, n = # of features, X = [m,n]
- Multiple Linear Regression Model Hypothesis:

$$f_{(w,b)}(X) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + b = np.dot(x, w) + b$$

- Python numpy vectorization perform addition and multiplication in parallel
- Vectorization is efficient as it scales to large data calculation as follows

$$X = [m, n], Y = [m, 1], W = [n, 1], b$$

$$Hypothesis\colon\ f_{wb} = np.dot(X, W) + b, \qquad Cost\colon J_{wb} = \frac{np.sum\left((f_{wb} - Y)^2\right)}{2m}$$

$$Derivates\colon dj_{db} = \frac{np.sum(f_{wb} - Y)}{m}, dj_{wb} = \frac{np.dot\left(X.T, (f_{wb} - Y)\right)}{m}$$

DeepLearning.AI Stanford ONLINE

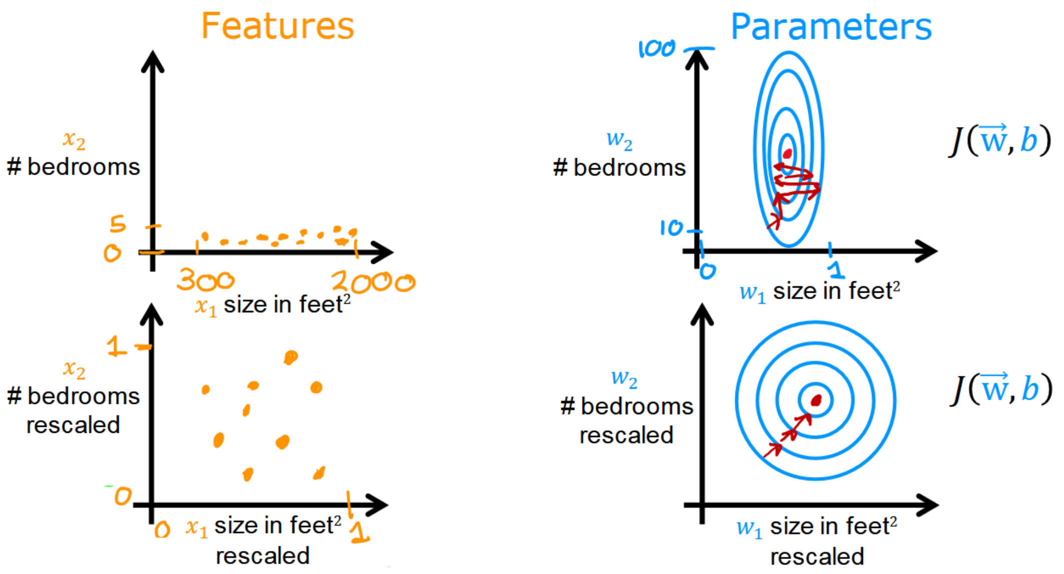
FEATURE SCALING

- Normal Equation: Closed form equation to solve the weights of Linear Regression

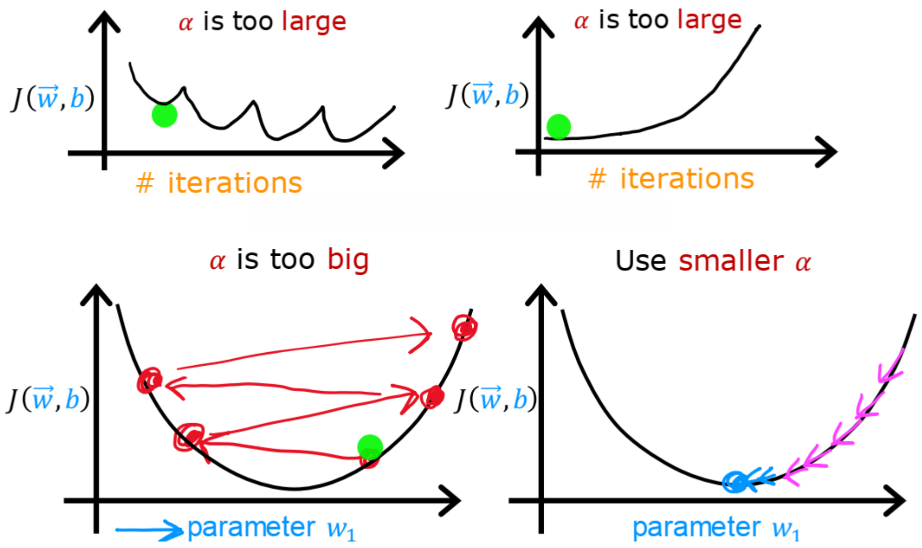
$$Weights:\ W = (X^T X)^{-1}.(X^T Y), \qquad Cost:\ f_{wb} = \frac{(XW - Y)^T (XW - Y)}{2m}$$

- Normal Equation solution runs slow when number of feature n is large
- Feature Scaling: Scale features to plot uniformly in specific or comparable range which makes the contour plot of features a circle. Scaling prevents feature bias and helps converge faster. Scale large feature values comparable to other smaller features.
- Scale feature with max value yield $0 < \frac{x}{maxValue} < 1$, acceptable range [-3, 3]
- Mean Normalization: Scale features $\frac{x - \mu}{\max - min}$ to center around zero with in range -1 and +1
- Z-score Normalization: Scale features $\frac{x - \mu}{\sigma}$ to center around zero with in range -3 and +3

LINEAR REGRESSION



Feature Scaling



Learning Rate

LINEAR REGRESSION

- Gradient Descent Convergence: Learning Curve plot of cost J in terms of iteration shows cost J decreasing over iteration.
- Auto converge test can be performed by check decrease of cost J less than constant $\epsilon = 0.001$
- Learning curve creates wiggle or diverge or increase of cost J if learning rate $\boldsymbol{\alpha}$ is very large
- Choose values of learning rate $\boldsymbol{\alpha}$ from 0.001 to 1 increasing by 3 fold as 0.001, 0.003, 0.01, 0.03, 0.1
- Feature Engineering: Choose custom features by transforming or combining original features

  EX: Take area as a new feature for predicting housing price from given input length and width
- Polynomial Regression: Polynomial function of 2nd, 3rd, 4th or 5th degree is chosen as hypothesis to fit curve or non-linear function for the input data

$$f_{(w,b)}(X) = w_1x_1 + w_2{x_2}^2 + w_3{x_3}^3 + b \text{ or } f_{(w,b)}(X) = w_1x_1 + w_2\sqrt{x_2} + b$$

- Feature scaling is needed for polynomial regression to keep features in comparable range